A Design Framework for Hardware-Efficient Logarithmic Floating-Point Multipliers

Tingting Zhang, *Graduate Student Member, IEEE,* Zijing Niu, and Jie Han, *Senior Member, IEEE*

Abstract—The symbiotic use of logarithmic approximation in floating-point (FP) multiplication can significantly reduce the hardware complexity of a multiplier. However, it is difficult for a limited number of logarithmic FP multipliers (LFPMs) to fit in a specific error-tolerant application, such as neural networks (NNs) and digital signal processing, due to their unique error characteristics. This paper proposes a design framework for generating LFPMs. We consider two FP representation formats with different ranges of mantissas, the IEEE 754 Standard FP Format and the Nearest Power of Two FP Format. For both logarithm and anti-logarithm computation, the applicable regions of inputs are first evenly divided into several intervals, and then approximation methods with negative or positive errors are developed for each sub-region. By using piece-wise functions, different configurations of approximation methods throughout applicable regions are created, leading to LFPMs with various trade-offs between accuracy and hardware cost. The variety of error characteristics of LFPMs is discussed and the generic hardware implementation is illustrated. As case studies, two LFPM designs are presented and evaluated in applications of JPEG compression and NNs. They do not only increase the classification accuracy, but also achieve smaller PDPs compared to the exact FP multiplier, while being more accurate than a recent logarithmic FP design.

Index Terms—Floating-point multiplier, logarithmic multiplier, neural networks, JPEG compression, error tolerance, approximate computing, approximate multiplier.

1 Introduction

Due to the dynamic range and high accuracy in numerical representation, floating-point (FP) arithmetic is widely used in applications where the magnitude of a value plays an important role [1]. For example, an FP representation is usually adopted in the training process of neural networks (NNs) for its more accurate encoding of weights and activations [2]. In the image processing domain, it is usually deployed in processing cinematic materials [3] and representing pixel information in modern shading units in graphics processing units [4]. However, FP arithmetic units require a substantial amount of power and area. Moreover, as the Dennard scaling is coming to an end, the increase in power density has become a limitation to further improve the performance.

As an emerging computing paradigm, approximate computing improves the efficiency of circuits and systems at the cost of limited precision for error-tolerant applications, such as digital signal processing (DSP), data mining, and NNs [5], [6]. As a basic operation, multiplication is interesting for approximation due to its high cost incurred in circuit implementation [7]–[9]. Approximate FP multiplier designs have been investigated by using approximate Booth encoding [10], reconfiguration [7], [11], truncation techniques [12] and approximate adders [8], [13]. However, these methods lead to limited performance improvement since the hardware-consuming multiplication can not be totally eliminated.

Logarithmic computation has been considered in training deep NNs [14] and DSP [15] for recent years. Moreover, the attempts to apply logarithmic multiplication in Lognet lead to higher classification accuracy at low resolutions [16]. Logarithmic computing provides an energy-efficient alternative for multiplication, which converts conventional multiplication to addition. Since its number representation is naturally suited to FP numbers, efficient logarithmic FP multipliers (LFPMs) have been pursued to reduce circuit area and improve computational efficiency [17]–[21]. The design in [20], which always underestimates the product, significantly reduces circuit area and power. However, the accumulated negative errors in the training process lead to catastrophic results. An LFPM with double-sided errors [21] was developed using the Nearest Power of Two FP Format (FPNP2) to provide a more accurate result at a cost of relatively large circuits.

Although a number of approximate fixed-point multipliers can be found in the literature [22], [23], current approximate FP multiplier designs are not sufficient. Due to the unknown error characteristics in applications, it is difficult to take advantage of a trade-off between the application-level performance and hardware efficiency. Hence, additional approximate designs with a diverse range of accuracies and circuit characteristics are required to gain a better understanding of their roles in different applications. In this paper, we pursue the design of LFPMs for error-tolerant applications that require dynamic numerical representation for computation.

This paper presents a design framework to generate a library of LFPMs. The proposed LFPMs are developed by using two FP representation formats, the IEEE 754 Standard FP Format (FP754) and FPNP2, in the logarithm and anti-logarithm approximation. In both processes, the applicable regions of inputs are first evenly divided into several sub-

• *T. Zhang, Z. Niu and J. Han are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada.*
E-mail: ttzhang@ualberta.ca, zijing2@ualberta.ca, jhan8@ualberta.ca.

regions. Then, rather than using a single approximation method, two candidate designs that respectively introduce negative and positive errors are considered for each sub-region. Numerous LFPMs are generated by configuring different approximation methods throughout the regions based on a piece-wise function. The error analysis and hardware implementations are presented. The error sensitivities of applications for using approximate FP multipliers are analyzed by replacing the FP multiplication with the generated LFPMs. An appropriate LFPM design can then be selected for a good trade-off between accuracy and hardware efficiency. Two example designs are synthesized and used in JPEG compression and NN training applications as case studies and for evaluation.

The remainder of this paper is organized as follows. Section 2 presents the basics. Section 3 discusses the approximation methods. The generic hardware implementation of LFPMs are analyzed in Section 4. Case studies and applications in JPEG compression and NN training are discussed in Section 5. Finally, Section 6 concludes this paper.

2 Preliminaries

2.1 FP Representation

2.1.1 IEEE 754 Standard FP Format (FP754)

As defined in the IEEE 754 Standard, the FP number is represented as a string that consists of a 1-bit sign S, a w-bit exponent E and a q-bit mantissa M. An FP number (denoted as N) is expressed to as [24]

$$N = (-1)^S \cdot 2^{E-bias} \cdot (1+x), \tag{1}$$

where S takes either 0 or 1 for a positive number or a negative number, respectively. The exponent has a $bias$ of $2^{w-1}-1$ to ensure $E \geq 0$, thus the actual exponent is given by $E - bias$. The actual mantissa includes a hidden '1' (i.e., $1.M$, also referred as $X(=1+x)$, where $0 \leq x < 1$).

2.1.2 Nearest Power of Two FP Format (FPNP2)

The FP754 provides the largest power of two smaller than N in the exponent E. Based on it, another FP format, named the Nearest Power of Two, was considered in an LFPM design [21], [25]. It converts the exponent to obtain the nearest power of two to N by comparing the mantissa x with 0.5, as

$$N = (-1)^S \cdot 2^{E^{'}-bias} \cdot (1+x^{'}), \tag{2}$$

where

$$E^{'} = \begin{cases} E, & x < 0.5 \\ E+1, & x \geq 0.5 \end{cases}, \tag{3}$$

$$X^{'} = 1 + x^{'} = \begin{cases} 1+x, & x < 0.5 \\ \frac{1+x}{2}, & x \geq 0.5 \end{cases}, \tag{4}$$

where $-0.25 \leq x^{'} < 0.5$ and $0.75 \leq X^{'} < 1.5$.

2.2 Logarithmic FP Multiplication

Consider $P = A \times B$, the FP multiplication consists of the sign bit generation, the addition of the exponents, and the multiplication of the mantissas. Let the sign bits, exponents, and mantissas of A, B and P be respectively denoted with the corresponding subscripts. Assume two input operands be represented by using FP754 or FPNP2 and the final multiplication result P be represented by using FP754. The FP754 and FPNP2 are different with each other in mantissa and exponent expressions. The exponents, and mantissas of A, B are denoted by $\hat{x}$ (can be x in FP754 or $x^{'}$ in FPNP2) and $\hat{E}$ (can be E in FP754 or $E^{'}$ in FPNP2), respectively. The logarithmic FP multiplication is given by [21]

$$S_p = S_A \oplus S_B, \tag{5}$$

$$X_{AB} = \hat{X}_A \hat{X}_B = (1+\hat{x}_A) \times (1+\hat{x}_B), \tag{6}$$

$$E_P = \begin{cases} \hat{E}_A + \hat{E}_B - bias - 1, & X_{AB} < 1 \\ \hat{E}_A + \hat{E}_B - bias, & 1 \leq X_{AB} < 2 \\ \hat{E}_A + \hat{E}_B - bias + 1, & otherwise \end{cases}, \tag{7}$$

$$X_P = \begin{cases} 2X_{AB}, & X_{AB} < 1 \\ X_{AB}, & 1 \leq X_{AB} < 2 \\ \frac{X_{AB}}{2}, & otherwise \end{cases}, \tag{8}$$

where the exponent and mantissa of the product P are determined by the comparison of the obtained mantissa X_{AB} with 1 and 2.

To ease the complexity of computing X_{AB}, logarithmic multiplication in base-2 scientific notation converts the multiplication in (6) to addition as

$$log_2 X_{AB} = log_2(1+\hat{x}_A) + log_2(1+\hat{x}_B). \tag{9}$$

Then, the logarithmic result ($log_2 X_{AB}$) is interpolated back into the original result (X_{AB}) using anti-logarithm methods. Finally, the E_P and X_P are obtained by using (7) and (8).

Mitchell's logarithm approximation (LA) and anti-logarithm approximation (ALA) are commonly used as [26]

$$log_2(1+k) \cong k, \tag{10}$$

where $0 \leq k < 1$.

$$2^l \cong l + 1, \tag{11}$$

where $0 \leq l < 1$.

2.3 Error Metrics

In this paper, four error metrics are considered to evaluate the approximate designs, i,e., the mean error distance (MED), the mean absolute error distance ($MAED$), the mean relative error distance ($MRED$) and the mean absolute relative error distance ($MARED$) [27] [28]. The MED and the $MAED$ are the averages of the signed difference and the absolute difference between the approximate and the exact results, respectively. The $MRED$ and the $MARED$ are defined as the averages of the signed difference and absolute difference divided by the exact result, respectively. The $MAED$ and $MARED$ computed by absolute errors indicate the magnitude difference in the multiplication result. The $MAED$ measures the absolute discrepancy between the approximate multiplier and the accurate multiplier, whereas the $MARED$ assesses their absolute relative discrepancy, which is scale-independent. The MED and $MRED$ consider the offset effect of underestimation and overestimation errors that occur in the accumulation. Therefore, these four error metrics can show the general error behavior of a multiplier design.

3 Approximation Design Framework

3.1 Logarithm Approximation (LA)

3.1.1 *Proposed Method*

The input operands for the logarithm approximation use either the FP754 (given by x_A and x_B) or the FPNP2 (given by x'_A and x'_B) formats. When using the FP754, the LA for $log_2(1+k)$ is considered for $0 \le k < 1$; when using the FPNP2 as in (2), the LA is considered for $-0.25 \le k < 0.5$.

Mitchell's LA by (10) always underestimates the logarithm when $0 \le k < 1$, which introduces negative errors. However, beyond the applicable region, (10) overestimates the logarithm. Hence, by applying Mitchell's LA method out of the original applicable region, positive errors are introduced.

Firstly, we extend Mitchell's LA in (10) into the region out of $0 \le k < 1$. Considering that the multiplication or division of powers of two can be easily performed by shifts, we assume a number, to be n, is a power of two. Let $-\infty < k < +\infty$, the LA methods can be obtained from (10) with different applicable regions, as

$$\begin{aligned} log_2(1+k) &= log_2(n \times \tfrac{1+k}{n}) \\ &= log_2(n) + log_2(1 + \tfrac{1+k}{n} - 1) \\ &\cong log_2(n) + \tfrac{1+k}{n} - 1, \end{aligned} \tag{12}$$

where $0 \le \frac{1+k}{n} - 1 < 1$, thus $n-1 \le k < 2n-1$.

Consider both FP754 (where $0 \le k < 1$) and FPNP2 (where $-0.25 \le k < 0.5$). The LA methods in which applicable regions are close to the range of $[-0.25, 1)$ are considered. Let $n = \frac{1}{2}$, 1, and 2, then

$$log_2(1+k) \cong \begin{cases} 2k, & -0.5 \le k < 0 \\ k, & 0 \le k < 1 \\ \frac{k+1}{2}, & 1 \le k < 3 \end{cases} . \tag{13}$$

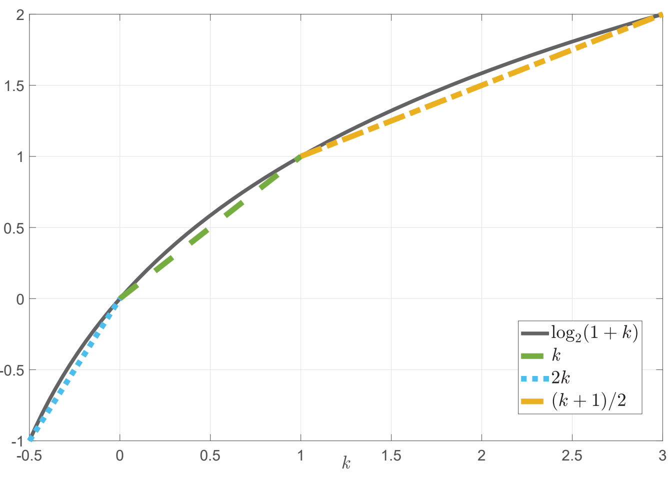


Fig. 1: LA in (13).

TABLE 1: Piece-wise Logarithm Approximation using IEEE 754 Standard FP format

k		$[0, 0.25)$	$[0.25, 0.5)$	$[0.5, 0.75)$	$[0.75, 1)$
$log_2(1+k)$	positive	$2k$	$\frac{k+1}{2}$	$\frac{k+1}{2}$	$\frac{k+1}{2}$
	negative	k	k	k	k

TABLE 2: Piece-wise Logarithm Approximation using Nearest Power of Two FP Format

k		$[-0.25, 0)$	$[0, 0.25)$	$[0.25, 0.5)$
$log_2(1+k)$	positive	k	$2k$	$\frac{k+1}{2}$
	negative	$2k$	k	k

Fig. 1 indicates that these LA methods introduce negative errors within their respective applicable regions; otherwise, they overestimate the logarithm results. The proposed LA method is developed by considering three candidates (i.e., k, $\frac{k+1}{2}$ and $2k$) for $-0.25 \le k < 1$ to introduce double-sided errors. When using FP754, i.e., $0 \le k < 1$, $2k$ and $\frac{k+1}{2}$ overestimate the results, whereas k underestimates the results. When using FPNP2, i.e., $-0.25 \le k < 0.5$, k, $2k$ and $\frac{k+1}{2}$ overestimate the results in the regions of $[-0.25, 0)$, $[0, 0.5)$ and $[0, 0.5)$, respectively. Moreover, $2k$ and k underestimate the results in the regions of $[-0.25, 0)$ and $[0, 0.5)$, respectively.

The LAs for $log_2(1+k)$ when $0 \le k < 1$ based on FP754 and $-0.25 \le k < 0.5$ based on FPNP2 are both developed by using a piece-wise function with different configurations in these sub-regions, respectively. First, the region is divided into several sub-regions with a fixed width. Then, in each sub-region, two candidates with a higher accuracy are considered, one overestimating the logarithmic result with positive errors and another underestimating the logarithmic result with negative errors. An LA method can be independently selected in each single sub-region.

A trade-off is assessed when deciding an appropriate length of each sub-region. An excessively small length leads to an increase of hardware complexity, whereas an excessively large length reduces the variety of the accuracy for those generated designs. If 0.25 is taken as the length of each sub-region, the piece-wise logarithm approximation (PWLA) based on FP754 and FPNP2 are presented in Tables 1 and 2, respectively.

3.1.2 *Theoretical Analysis*

Assume the approximate logarithm result be $\alpha_i(k)$, where $-0.25 \le k < 1$ and i ($\in \{1, 2, 3\}$) indicates a label to distinguish different LA methods. The error in the logarithm approximation, $\varepsilon_i(k)$, is given by

$$\varepsilon_i(k) = \alpha_i(k) - log_2(1+k), \tag{14}$$

where $\alpha_1(k) = 2k$, $\alpha_2(k) = k$, and $\alpha_3(k) = \frac{k+1}{2}$.

To compare the accuracy of two candidate LA methods which overestimates or underestimates the logarithmic results in each sub-region, the difference between the $|\varepsilon_i(k)|$ of two LA methods, denoted by D_1, is computed by

$$D_1 = \begin{cases} |\varepsilon_2(k)| - |\varepsilon_1(k)| = 3k - 2log_2(1+k), -0.25 \le k < 0 \\ |\varepsilon_1(k)| - |\varepsilon_2(k)| = 3k - 2log_2(1+k), 0 \le k < 0.25 \\ |\varepsilon_3(k)| - |\varepsilon_2(k)| = \frac{3k+1}{2} - 2log_2(1+k), 0.25 \le k < 1 \end{cases} \tag{15}$$

Fig. 2 plots two candidate LA methods in each sub-region based on FP754 or FPNP2 in Table 1. The errors for different LA methods in four sub-regions are analyzed as follows.

$-0.25 \le k < 0$: The errors decrease with k when using $\alpha_2(k)$ and $\alpha_1(k)$ for LA. Thus, $\varepsilon_2(k)$ and $\varepsilon_1(k)$ are in the range of $[0, 0.1650)$ and $[-0.0849, 0)$, respectively. D_1 is always larger than or equal to 0, thus $\alpha_2(k) = k$ is less accurate than $\alpha_1(k) = 2k$.

$0 \le k < 0.25$: The errors for the use of $\alpha_2(k)$ and $\alpha_1(k)$ increase with k. Thus, $\varepsilon_2(k)$ and $\varepsilon_1(k)$ are in the range of $(-0.0719, 0]$ and $[0, 0.1780)$, respectively. D_1 is always larger than or equal to 0, indicating that $\alpha_2(k) = k$ is more accurate than $\alpha_1(k) = 2k$.

$0.25 \le k < 0.5$: The use of $\alpha_2(k)$ first increases errors and then decreases errors with k, whereas the use of $\alpha_3(k)$ decreases the errors with k. For $\alpha_2(k)$, the maximum absolute error of 0.0863 can be found at $k = 0.4426$ by differentiating $\varepsilon_2(k)$ and setting it equal to zero, as $\frac{d\varepsilon_2(k)}{dk} = 0$ [26]. $\varepsilon_2(k)$ and $\varepsilon_3(k)$ are in the range of $[-0.0863, -0.0719]$ and $(0.1650, 0.3030]$, respectively. $\alpha_2(k) = k$ is obviously more accurate than $\alpha_3(k) = \frac{k+1}{2}$ for LA.

$0.5 \le k < 0.75$: The errors due to the use of LA methods of $\alpha_2(k)$ and $\alpha_3(k)$ decrease with k. $\varepsilon_2(k)$ and $\varepsilon_3(k)$ are in the range of $[-0.0849, -0.0573)$ and $(0.0676, 0.1650]$, respectively. D_1 is always larger than or equal to 0, thus $\alpha_2(k) = k$ is the more accurate LA method.

$0.75 \le k < 1$: The errors of using $\alpha_2(k)$ and $\alpha_3(k)$ decrease with k. $\varepsilon_2(k)$ and $\varepsilon_3(k)$ are in the range of $[-0.0573, 0)$ and $(0, 0.0676]$, respectively. Let D_1 be less than or equal to 0, then $0.8491 \le k < 1$. Therefore, $\alpha_2(k) = k$ and $\alpha_3(k) = \frac{k+1}{2}$ are more accurate when $0.75 \le k < 0.8491$ and $0.8491 \le k < 1$, respectively.

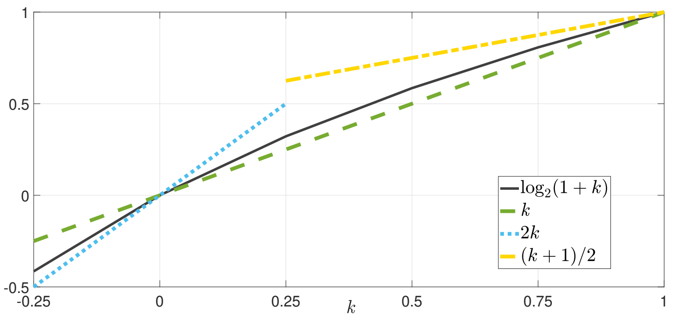


Fig. 2: Piece-wise logarithm approximation functions.

3.2 Anti-logarithm Approximation (ALA)

3.2.1 Proposed Method

Consider that two logarithm results in the LA process can be obtained from the approximation based on either FP754 or FPNP2, as in Tables 1 and 2. Then, the sum of two logarithm results, denoted by l, ranges from -1 to 2, so the ALA method is considered for $-1 \le l < 2$. The ALA method as per (11) always introduces positive errors in the applicable region. Similar as the methods in Section 3.1, the ALA methods can be derived from (11), as

$$2^l = n \times 2^{l - log_2 n} \cong n \times (l + 1 - log_2 n), \tag{16}$$

TABLE 3: Piece-wise Anti-logarithm Approximation

	l	$[-1,-0.5)$	$[-0.5,0)$	$[0,0.5)$	$[0.5,1)$	$[1,1.5)$	$[1.5,2)$
2^l	positive	$\frac{l+2}{2}$	$\frac{l+2}{2}$	$l+1$	$l+1$	$2l$	$2l$
	negative	$\frac{l+3}{4}$	$l+1$	$\frac{l+2}{2}$	$2l$	$l+1$	$4l-4$

where $0 \le l - log_2 n < 1$, thus $log_2 n \le l < log_2 n + 1$.

We consider the ALA methods in which applicable regions are close to the range of $[-1, 2)$. Let $n = \frac{1}{4}, \frac{1}{2}, 1, 2$, and 4, ALA methods can be obtained as

$$2^l \cong \begin{cases} \frac{l+3}{4}, & -2 \le l < -1 \\ \frac{l+2}{2}, & -1 \le l < 0 \\ l+1, & 0 \le l < 1 \\ 2l, & 1 \le l < 2 \\ 4l-4, & 2 \le l < 3 \end{cases}. \tag{17}$$

As shown in Fig. 3, these ALA methods always overestimate the results, whereas negative errors are introduced out of the applicable regions.

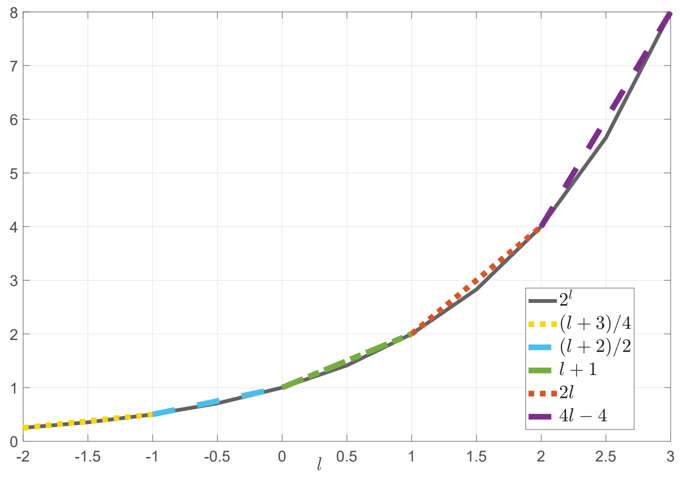


Fig. 3: ALA in (17).

The proposed ALA method is developed based on five candidate ALA methods (using $\frac{l+3}{4}$, $\frac{l+2}{2}$, $l+1$, $2l$ and $4l-4$) for $-1 \le l < 2$ to introduce double-sided errors. Similar to the PWLA method, two approximation methods are considered in each sub-region, one with positive errors and another with negative errors. The piece-wise anti-logarithm approximation (PWALA) method takes 0.5 as the width of each sub-region, as per Table 3.

3.2.2 Theoretical Analysis

Let the approximate anti-logarithm result be $\beta_j(l)$, where $-1 \le l < 2$ and j ($\in \{1, 2, 3, 4, 5\}$) indicates a label to distinguish different ALA methods. The error denoted by $\varepsilon_j(l)$ is given by

$$\varepsilon_j(l) = \beta_j(l) - 2^l, \tag{18}$$

where $\beta_1(l) = \frac{l+3}{4}$, $\beta_2(l) = \frac{l+2}{2}$, $\beta_3(l) = l+1$, $\beta_4(l) = 2l$, and $\beta_5(l) = 4l-4$.

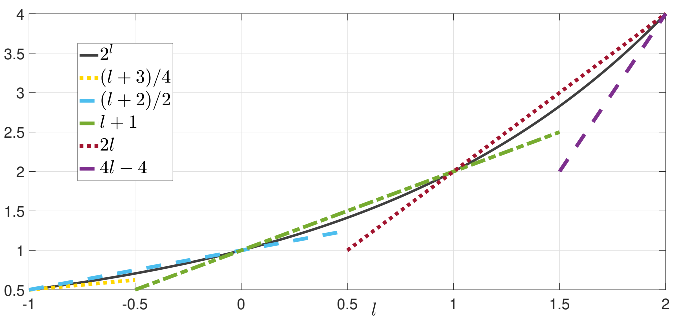


Fig. 4: Piece-wise anti-logarithm approximation functions.

Similarly, the difference between $|\varepsilon_j(l)|$ of two candidate ALA methods in each sub-region, denoted by D_2, is given by

$$D_2 = \begin{cases} |\varepsilon_2(l)| - |\varepsilon_1(l)| = \frac{3l+7}{4} - 2^{l+1}, & -1 \le l < -0.5 \\ |\varepsilon_2(l)| - |\varepsilon_3(l)| = \frac{3l+4}{2} - 2^{l+1}, & -0.5 \le l < 0 \\ |\varepsilon_3(l)| - |\varepsilon_2(l)| = \frac{3l+4}{2} - 2^{l+1}, & 0 \le l < 0.5 \\ |\varepsilon_3(l)| - |\varepsilon_4(l)| = 3l+1 - 2^{l+1}, & 0.5 \le l < 1 \\ |\varepsilon_4(l)| - |\varepsilon_3(l)| = 3l+1 - 2^{l+1} & 1 \le l < 1.5 \\ |\varepsilon_4(l)| - |\varepsilon_5(l)|| = 6l+4 - 2^{l+1} & 1.5 \le l < 2 \end{cases} . \quad (19)$$

The PWALA methods in Table 3 are presented in Fig. 4. Similarly, the errors of using different ALA methods in each sub-region are discussed as follows.

$-1 \le l < -0.5$: The errors for using $\beta_1(l)$ and $\beta_2(l)$ increase with l. Thus, $\varepsilon_1(l)$ and $\varepsilon_2(l)$ are in the range of $(-0.0821, 0]$ and $[0, 0.0428)$, respectively. D_2 is always less than or equal to 0. Therefore, $\beta_2(l) = \frac{l+2}{2}$ is more accurate.

$-0.5 \le l < 0$: The error introduced by the use of $\beta_3(l)$ decreases with l, whereas that for using $\beta_2(l)$ first increases and then decreases with l. The maximum error of 0.0430 for using $\beta_2(l)$ can be achieved when $l = \frac{ln\frac{1}{2ln2}}{ln2} \approx -0.471$. Thus, $\varepsilon_2(l)$ and $\varepsilon_3(l)$ are in the range of $(0, 0.0430]$ and $(-0.2071, 0]$, respectively. D_2 is always less than or equal to 0, indicating $\beta_2(l) = \frac{l+2}{2}$ is more accurate.

$0 \le l < 0.5$: The errors introduced by the use of $\beta_2(l)$ and $\beta_3(l)$ increase with l. Thus, $\varepsilon_2(l)$ and $\varepsilon_3(l)$ are in the range of $(-0.1642, 0]$ and $[0, 0.0857)$, respectively. D_2 is always less than or equal to 0 when $0.2245 \le l < 0.5$. Therefore, $\beta_2(l) = \frac{l+2}{2}$ and $\beta_3(l) = l+1$ are more accurate when $0 \le l < 0.2245$ and $0.2245 \le l < 0.5$, respectively.

$0.5 \le l < 1$: The error for using $\beta_4(l)$ decreases with l, whereas the error for using $\beta_3(l)$ first increases and then decreases with l. The maximum error of 0.0860 for using $\beta_3(l)$ can be achieved when $l = \frac{ln\frac{1}{ln2}}{ln2} \approx 0.5287$. $\varepsilon_4(l)$ and $\varepsilon_3(l)$ are in the ranges of $[-0.4142, 0)$ and $(0, 0.0860]$, respectively. D_2 is always smaller than 0, thus $\beta_3(l) = l+1$ is more accurate.

$1 \le l < 1.5$: The errors for the use of $\beta_4(l)$ and $\beta_3(l)$ increase with l. $\varepsilon_4(l)$ and $\varepsilon_3(l)$ are in the range of $[0, 0.1715)$ and $(-0.3284, 0]$, respectively. D_2 is less than or equal to 0 when $1.2245 \le l < 1.5$. Thus, $\beta_4(l) = 2l$ and $\beta_3(l) = l+1$ are more accurate when $1.2245 \le l < 1.5$ and $1 \le l < 1.2245$, respectively.

$1.5 \le l < 2$: The error for using $\beta_5(l)$ decreases with l, whereas the error for using $\beta_4(l)$ first increases and then decreases with l. The maximum error for using $\beta_4(l)$ is obtained by $\frac{d\varepsilon_4(l)}{dl} = 0$, where $l = \frac{ln\frac{2}{ln2}}{ln2} \approx 1.5287$, thus $\varepsilon_4 = 0.1721$. $\varepsilon_5(l)$ and $\varepsilon_4(l)$ are in the range of $[-0.8284, 0)$ and $(0, 0.1721]$, respectively. D_2 is always smaller than or equal to 0, thus $\beta_4(l) = 2l$ is the more accurate ALA method.

3.3 Logarithmic FP Multiplication

The logarithmic FP multiplication based on piece-wise approximation is developed from approximation in logarithm and anti-logarithm processes, as shown in Fig. 5.

Consider the LA process. F indicates the use of FP754 or FPNP2. The $\boldsymbol{C_{x_A}}$ (or $\boldsymbol{C_{x_B}}$) contains four elements, each taking either 'p' or 'n' to indicate using the approximation method with positive or negative errors in each interval. The use of FPNP2 converts x_A in the domain of $[0.5, 1)$ to $[-0.25, 0)$, resulting in the same configuration for $C_{x_A}(2)$ and $C_{x_A}(3)$ (or as $C_{x_B}(2)$ and $C_{x_B}(3)$). When x_A (or x_B) locates within $[0, 0.5)$, no matter which FP format is used, different LA methods are the same. When x_A (or x_B) is in the range of $[0.5, 1)$, FPNP2 modifies E_A (or E_B) to $E_A + 1$ (or $E_B + 1$) and perform the LA using either $\frac{x_A - 1}{2}$ or $x_A - 1$. Consider the ALA process. The $\boldsymbol{C_{a_l}}$ has six elements, each with either 'p' or 'n' . Firstly,X_{AB} is determined by the configuration for PWALA methods. Then, E_P and X_P are obtained to satisfy the requirements of the FP754 format depending on X_{AB}, as in (7)-(8).

There are 5744 piece-wise based LFPM (PWLM) designs generated using the proposed approximation method in total. The PWLM approximately computes logarithm and antilogarithm using linear equations. The regions of $0 \le x_A, x_B < 1$ are divided into γ sub-regions. Then, for the ith sub-region ($1 \le i \le \gamma$), it essentially approximates X_{AB} in (6) by $\widetilde{X}_{AB}$ using a linear equation $f(x_A, x_B)$ related to x_A and x_B, as

$$X_{AB} \cong \widetilde{X}_{AB} = f_i(x_A, x_B) = u_i \cdot x_A + v_i \cdot x_B + c_i \quad (20)$$

where values of coefficients u_i, v_i and c_i depend on the configurations of approximation methods in the dedicated PWLM.

Different configurations of PWLA methods for x_A and x_B, and PWALA methods can lead to possible error compensation. Therefore, the respective accuracy of LA and ALA methods can not predict the overall error characteristics of PWLMs. For single-precision LFPMs, samples of 10^7 random cases with a standard normal distribution (for floating numbers) and with uniform distribution (for the mantissa) were generated to obtain the error results. The PWLMs using two-piece ALA are more likely to introduce large errors. Three- and four-piece LA provides a diverse accuracy, whereas one- and two-piece LA leads to moderate accuracy.

Table 4 evenly divides the ranges of error results into five parts. Thus, the accuracy levels in four error metrics, i.e., $|MED|$, $|MRED|$, $MAED$, $MARED$, can be distinguished for each PWLM. The error results in $|MED|$ and $|MRED|$ obtained by uniformly (or normally) distributed random numbers range from 2.44×10^{-5} to 0.48 (or from 3×10^{-6} to 1.31×10^{-4}) and from 7.36×10^{-5} to 0.22 (or from 1.7×10^{-5} to 0.232) , respectively. Moreover, $MAED$ and $MARED$ are diverse in the range of $[0.06, 0.6]$ (or $[1.82 \times 10^{-2}, 0.19]$) and $[0.02, 0.29]$ (or $[2.88 \times 10^{-2}, 0.31]$) ,

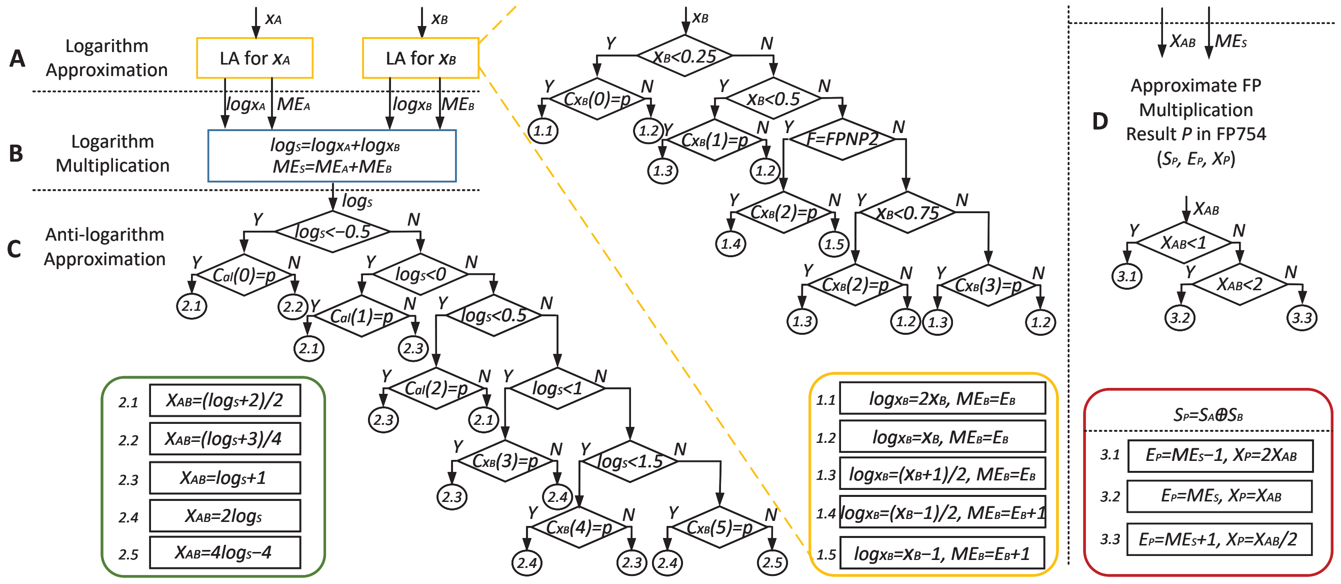


Fig. 5: Logarithmic FP multiplication based on piece-wise approximation. Inputs: FP format (F); Sign bits (S_A and S_B), exponents (E_A and E_B), and mantissas (x_A and x_B) in FP754; Configuration for logarithm approximation ($\boldsymbol{C_{x_A}}$ and $\boldsymbol{C_{x_B}}$); Configuration for anti-logarithm approximation ($\boldsymbol{C_{al}}$). Output: an approximate FP multiplication result P in FP754 (S_p, E_p, and X_p). ME_A and ME_B denote the modified exponents for A and B, respectively; log_{x_A} and log_{x_B} denote the logarithm mantissas for x_A and x_B, respectively; ME_s is the sum of ME_A and ME_B; log_s is the sum of log_{x_A} and log_{x_B}; X_{AB} is the anti-logarithm result of log_s.

TABLE 4: The Number of Single-precision PWLMs in Different Accuracy Levels of Four Error Metrics

Uniform Distribution					Normal Distribution				
$\|MED\|$					$\|MED\|$ ($\times10^{-4}$)				
$[2.44\times10^{-5}, 0.09)$	$[0.09, 0.19)$	$[0.19, 0.29)$	$[0.29, 0.38)$	$[0.38, 0.48]$	$[0.03, 0.29)$	$[0.29, 0.54)$	$[0.54, 0.80)$	$[0.80, 1.05)$	$[1.05, 1.31]$
2798	1813	810	268	55	2475	2881	299	75	14
$MAED$					$MAED$				
$[0.06, 0.17)$	$[0.17, 0.27)$	$[0.27, 0.38)$	$[0.38, 0.49)$	$[0.49, 0.60]$	$[0.02, 0.05)$	$[0.05, 0.09)$	$[0.09, 0.12)$	$[0.12, 0.16)$	$[0.16, 0.20]$
45	2204	2924	514	56	79	3183	2123	318	41
$\|MRED\|$					$\|MRED\|$				
$[7.36\times10^{-5}, 0.04)$	$[0.04, 0.09)$	$[0.09, 0.13)$	$[0.13, 0.18)$	$[0.18, 0.22]$	$[1.7\times10^{-5}, 0.04)$	$[0.04, 0.09)$	$[0.09, 0.14)$	$[0.14, 0.19)$	$[0.19, 0.23]$
2860	1803	779	255	47	2788	1847	812	250	47
$MARED$					$MARED$				
$[0.02, 0.08)$	$[0.08, 0.13)$	$[0.13, 0.18)$	$[0.18, 0.23)$	$[0.23, 0.29]$	$[0.02, 0.08)$	$[0.08, 0.14)$	$[0.14, 0.20)$	$[0.20, 0.25)$	$[0.25, 0.31]$
54	2794	2480	373	42	79	3200	2106	318	41

respectively. For the uniform distribution, the mean values of $|MED|$, $|MRED|$, $MAED$ and $MARED$ for these PWLMs are 0.1201, 0.0546, 0.2391 and 0.1097, respectively; and their variances are 0.0083, 0.0017, 0.0043 and 0.0009, respectively. For the standard normal distribution, the mean values of these four metrics are 0.1123, 0.0494, 0.2489 and 0.1151, respectively; and the variance are 0.0063, 0.0012, 0.0034 and 0.0006, respectively. There are around 80% of PWLMs with high accuracy levels (Levels 1 & 2), evaluated by $|MED|$ and $|MRED|$ and almost 90% of PWLMs locate in moderate accuracy levels (Levels 2 & 3), in terms of $MAED$ and $MARED$.

The error-tolerant applications differ from their error sensitivities of different error metrics. Various accuracies of our PWLM designs support a comprehensive error analysis when applying approximate FP multipliers to those applications.

4 Hardware Implementation

In this section, the generic circuit block diagram is introduced and the simplifications for each circuit block are investigated. Since the computation process can be simplified in different manners, the hardware implementation can be specially designed for each PWLM.

4.1 The Generic Circuit Block

Fig. 6 presents the generic circuit blocks for implementing these PWLMs. The sign S, the exponent E, and the mantissa M, of the FP number can be obtained directly. $1.M$ denotes the actual mantissa and is given as $M[q].M[q-1]M[q-2]\cdots M[1]M[0]$, where $M[q]$ denotes the hidden one.

The computation for the sign bit of the product, S_P, is implemented by an XOR gate with two input signs, S_A and S_B. To obtain the mantissa of the product, the approximate logarithmic values of $1.M_A$ and $1.M_B$, denoted as M'_A and M'_B respectively, are computed in the logarithm approximation block. The two obtained values are then summed

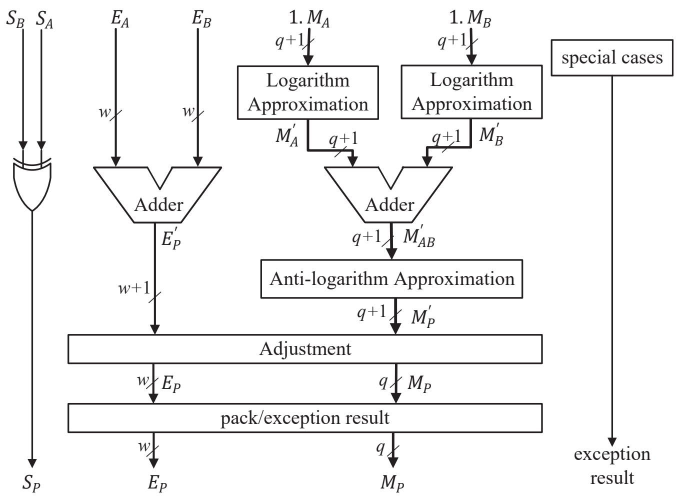


Fig. 6: The circuit block diagram for the generic implementation of PWLMs.

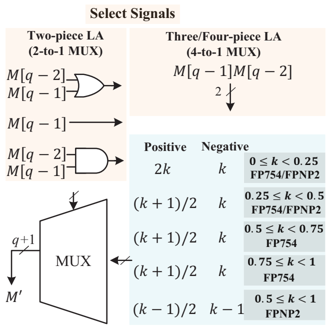
together since the multiplication is replaced by the addition in the logarithm domain. The sum of M'_A and M'_B, denoted by M'_{AB}, is propagated to the anti-logarithm approximation block to compute its approximate anti-logarithm, denoted by M'_P. The exponent of the product is firstly computed by the addition of E_A and E_B, denoted by E'_P. When the value of M'_P exceeds the range of the mantissa of the product, given by $[1, 2)$, M'_P and E'_P will be adjusted in the adjustment block. Finally, E_P and M_P are obtained. E_P is added with the bias to comply with FP754 format and the exception (such as overflow, underflow, and "not a number") is reported by detecting the preceding result. Note that the two operands are checked for exception at the beginning of the computation. Some circuit blocks in Fig. 6 can be combined into a single module and certain computational components can be implemented with less hardware complexity.

4.2 Logarithm Approximation Block

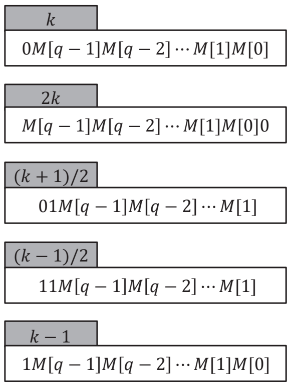
According to Table 1 and Table 2, the logarithm approximation block includes implementations for five different candidate LA methods and a multiplexer, as shown in Fig. 7, where M indicates M_A or M_B, and M' indicates M'_A or M'_B.

A multiplexer is used to implement the PWLA as shown in Fig. 7 (a). Particularly, when using one-piece LA (i.e., $\log_2(1+k) \cong k$), the multiplexer is not required. A 2-to-1 multiplexer is used for two-piece LA. The select signal is $M[q-2]$ OR $M[q-1]$, $M[q-1]$, or $M[q-1]$ AND $M[q-2]$, respectively when 0.25, 0.5 or 0.75 is the boundary value separating two sub-regions. Similarly, a 4-to-1 multiplexer is required for three-piece or four-piece LAs and the boundary value is determined by $M[q-1]M[q-2]$. For example, $M[q-1]M[q-2] = 11$ represents 0.75. The inputs of the multiplexer are selected from different LA methods in Table 1 and Table 2.

The five candidate LA methods using k, $2k$, $\frac{k+1}{2}$, $\frac{k-1}{2}$, and $k-1$ are implemented to serve as the inputs of the multiplexer as shown in Fig. 7 (b). Instead of directly implementing the equations, which would require adders and shifters, simple operations can be adopted for less hardware complexity. Since $1+k$ is implemented by $1.M[q-1]M[q-2]\cdots M[1]M[0]$, k is obtained as $0.M[q-1]M[q-2]\cdots M[1]M[0]$. The left/right shifter for implementing $\times 2$ and $/2$ operations is replaced by wire routing. Thus, $\frac{k+1}{2}$ and $2k$ are obtained as $0.M[q]M[q-1]M[q-2]\cdots M[1]$ and $M[q-1].M[q-2]\cdots M[1]0$, respectively. Note that $2k$ is used when $0 \leq k < 0.25$, and thus $M[q]$ is 0 and can be ignored after being shifted left. $\frac{k-1}{2}$ and $k-1$ are obtained as a negative number in 2's complement, respectively, i.e., $1.1M[q-1]\cdots M[1]$ and $1.M[q-1]\cdots M[0]$. Note that when PWLA based on FP754 (in Table 1) and FPNP2 (in Table 2) are both used for the two input operands, M' is implemented in $q+2$ bits instead of $q+1$ bits, where the sign bit of candidate LA methods in Fig. 7 (b) is extended by 1 bit.


Select Signals
Two-piece LA (2-to-1 MUX)
Three/Four-piece LA (4-to-1 MUX)
$M[q-2]$
$M[q-1]$
$M[q-1]M[q-2]$
2
$M[q-1]$
$M[q-2]$
$M[q-1]$
Positive
Negative
$2k$ k $0 \leq k < 0.25$ FP754/FPNP2
$(k+1)/2$ k $0.25 \leq k < 0.5$ FP754/FPNP2
$(k+1)/2$ k $0.5 \leq k < 0.75$ FP754
$(k+1)/2$ k $0.75 \leq k < 1$ FP754
$(k-1)/2$ $k-1$ $0.5 \leq k < 1$ FPNP2
$q+1$
MUX
M'

(a) For PWLA methods.


k
$0M[q-1]M[q-2]\cdots M[1]M[0]$
$2k$
$M[q-1]M[q-2]\cdots M[1]M[0]0$
$(k+1)/2$
$01M[q-1]M[q-2]\cdots M[1]$
$(k-1)/2$
$11M[q-1]M[q-2]\cdots M[1]$
$k-1$
$1M[q-1]M[q-2]\cdots M[1]M[0]$

(b) For candidate LA methods.

Fig. 7: Circuit designs for the logarithm approximation block.

4.3 Anti-logarithm Approximation Block

According to Table 3, a multiplexer and implementations of five candidate methods are required for computing the approximate anti-logarithm, M'_P, as shown in Fig. 8.

The PWALA is implemented by using a multiplexer, as shown in Fig. 8 (a). For the two-piece ALA, the select signal is $M'_{AB}[q-1]$ OR $M'_{AB}[q]$, $M'_{AB}[q]$, or $M'_{AB}[q]$ AND $M'_{AB}[q-1]$, respectively when 0.5, (1 or 0), or (1.5 or -0.5) is the boundary value separating two sub-regions. When using three-piece to five-piece ALAs, the boundary values are determined by $M'_{AB}[q]M'_{AB}[q-1]$, which is therefore the select signal for a 4-to-1 multiplexer. Particularly, when PWLAs based on FP754 (in Table 1) and FPNP2 (in Table 2) are both used for the two input operands, the PWALA method is applied to $-0.5 \leq l < 1.75$. M'_{AB} is implemented in $q+2$ bits with an extended sign bit, denoted by $M'_{AB}[q+1]$, to be distinguishable for the ranges of $[-0.5, 0)$ and $[1.5, 1.75)$. In this case, for using three-piece, four-piece and five-piece ALAs, $M'_{AB}[q+1]M'_{AB}[q]M'_{AB}[q-1]$ is used to determine the boundary values.

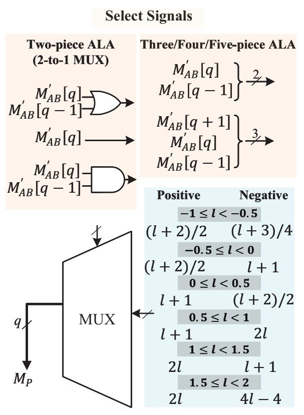
The four candidate ALA methods using $l+1$, $\frac{l+2}{2}$, $2l$ and $4l-4$, are implemented to obtain M'_P using wire routing. To simplify the circuit, the adjustment block is merged into the implementation of these ALA methods, where the adjusted value of M'_P, i.e., M_P, is used as the output for the multiplexer. The circuit to generate M_P is introduced in the next subsection.

4.4 Adjustment Block

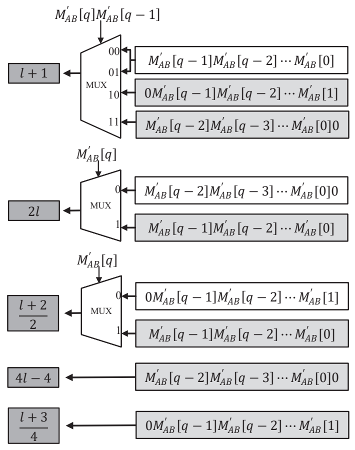
As shown in Fig. 8 (b), by merging the value adjustment into the ALA computation, the final mantissa, M_P, is directly implemented by M'_{AB} with a multiplexer in some cases.

According to Table 3, when $l \geq 1$, the values of $l+1$, $2l$ and $4l-4$ range from 2 to 4; when $l < 0$, the values of $l+1$, $\frac{l+2}{2}$ and $\frac{l+3}{4}$ range from 0.5 to 1. These values are divided by or multiplied by 2 to fit in the range of $[1, 2)$. Therefore, the value adjustment is determined by checking $M'_{AB}[q]M'_{AB}[q-1]$ for $l+1$ and $M'_{AB}[q]$ for other candidate ALA methods. The implementations for the adjusted ALA methods are shown as grey boxes in Fig. 8 (b). Particularly, $l+1$ is implemented for the range of $[-0.5, 0)$ when $M'_{AB}[q]M'_{AB}[q-1] = 11$ and for the range of $[1, 1.5)$ when $M'_{AB}[q]M'_{AB}[q-1] = 10$. $\frac{l+3}{4}$ and $4l-4$ are implemented without a multiplexer since they are used in one sub-region.

For the exponent, when ALA values are divided by or multiplied by 2, E'_P is added with or subtracted with one, respectively, to ensure correctness. When using PWLAs based on FP754 for the two inputs, to reduce the hardware complexity, a carry-in bit, implemented by $M'_{AB}[q]$, is used for the value adjustment of E'_P in the addition of E_A and E_B. When using PWLAs based on FPNP2, the final exponent is determined by both the FPNP2 conversion as per (3) and the value adjustment for the ALA values.



(a) For PWALA methods.

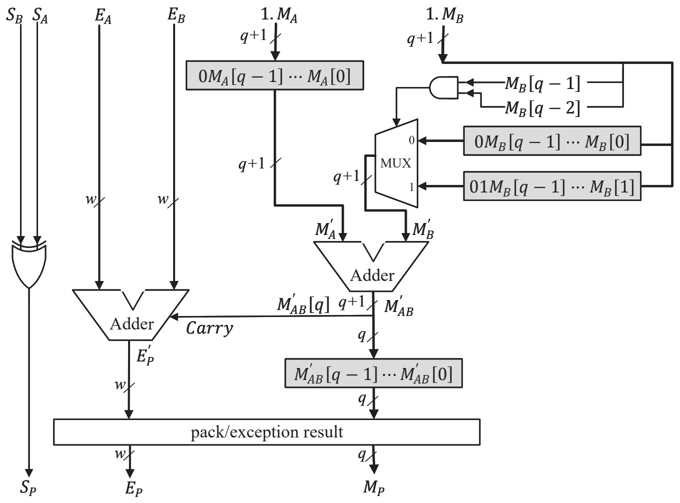


(b) For candidate ALA methods.

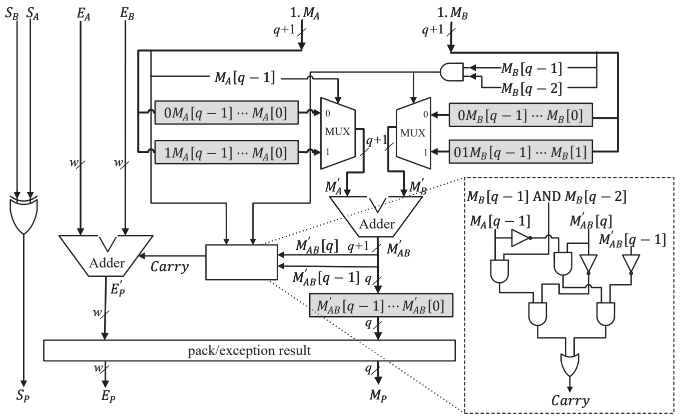
Fig. 8: Circuit designs for the anti-logarithm approximation block and the adjustment block.

5 Case Studies and Applications

The proposed PWLMs provide various accuracies and hardware performance, which can be used for error-tolerant applications with different requirements. To show the basic circuit design flow of the proposed PWLMs and assess their effectiveness, two example designs (PWLM1 and PWLM2) are studied in this section.



(a) For PWLM1.



(b) For PWLM2.

Fig. 9: The circuit designs for two example PWLMs.

5.1 Example PWLM Designs

For LA, PWLM1 approximates $log_2(1 + x_A)$ by using x_A, and approximates $log_2(1 + x_B)$ by using x_B when $x_B < 0.75$ and using $\frac{x_B+1}{2}$ otherwise. In PWLM2, $log_2(1 + x_A)$ is approximated by using x_A when $x_A < 0.75$ and $\frac{x_A+1}{2}$ otherwise. The input operand B is represented by FPNP2 and then $log_2(1 + x'_B)$ is approximated by using x'_B when $x_B < 0.5$ and $2x'_B$ otherwise. For ALA, let l be the sum of the approximate results $log_2(1 + x_A)$ and $log_2(1 + x_B)$. For PWLM1, $0 \leq l < 2$, thus it approximates 2^l by using $l + 1$ when $l < 1$ and using $2l$ otherwise. For PWLM2, $-0.5 \leq l < 1.5$, thus it approximates 2^l by using $\frac{l+2}{2}$ when $l < 0$, $l + 1$ when $0 \leq l < 1$, and $2l$ otherwise.

The circuit designs for PWLM1 and PWLM2 are shown in Fig. 9. By combining the value adjustment with the ALA computation block, the mantissas of final products after using a two-piece ALA in PWLM1 and a three-piece ALA in PWLM2 can both be simply implemented as $M'_{AB}[q-1]M'_{AB}[q-2]\cdots M'_{AB}[0]$, according to Fig. 8. For the exponent, due to the use of FPNP2-based LA, the carry-in bit for computing E'_P of PWLM2 is determined by the select signals of LAs, i.e., $M'_A[q-1]$ and $M'_B[q-1]M'_B[q-2]$, $M'_{AB}[q]$, and $M'_{AB}[q-1]$. The carry-in bit for PWLM1 is determined by the value adjustment for ALA and is

TABLE 5: Performance of FP Multipliers (Measured by Hardware and Error Metrics) and Applications in JPEG Compression (Measured by PSNR) and NN Training (Measured by Classification Accuracy)

	EFM	FPLM [21]	LAM [20]	**PWLM1**	**PWLM2**
AppM	-	N2-N2-2	I1-I1-2	I1-I2-2	I2-N2-3
Single-Precision					
Power (μW)	599.6	30.3	17.1	20.6	21.3
Area (μm^2)	2537.7	234.1	143.1	168.7	182.2
Delay (ns)	3.70	2.36	1.98	2.10	2.21
PDP (fJ)	2218.5	71.5	33.9	43.4	47.1
$\lvert MED\rvert$ ($\times 10^{-5}$)	0	0.83	1.24	0.96	1.18
$\lvert MRED\rvert$ ($\times 10^{-2}$)	0	0.14	3.82	2.95	2.95
$MAED$ ($\times 10^{-2}$)	0	1.82	2.42	2.39	2.39
$MARED$ ($\times 10^{-2}$)	0	2.88	3.82	3.78	3.78
PSNR (dB)	32.27	30.10	24.91	25.69	25.28
NN Accuracy (%)	97.99	98.03	97.99	98.01	**98.05**
Half-Precision					
Power (μW)	157.5	15.1	9.9	11.6	12.1
Area (μm^2)	868.8	119.9	81.2	97.5	101.3
Delay (ns)	3.03	1.25	0.97	1.10	1.21
PDP (fJ)	477.2	18.9	9.6	12.7	14.6
$\lvert MED\rvert$ ($\times 10^{-5}$)	0.018	0.83	1.26	1.20	1.21
$\lvert MRED\rvert$ ($\times 10^{-2}$)	0.13	0.15	3.67	2.80	2.80
$MAED$ ($\times 10^{-2}$)	0.067	1.83	2.46	2.42	2.42
$MARED$ ($\times 10^{-2}$)	0.33	3.11	4.10	4.03	4.03
PSNR (dB)	32.26	30.08	24.86	25.62	25.23
NN Accuracy (%)	96.46	96.15	96.03	**96.24**	95.84
Bfloat16					
Power (μW)	107.8	15.0	11.6	12.3	13.1
Area (μm^2)	724.6	123.3	96.4	101.4	106.6
Delay (ns)	2.90	1.23	0.97	0.99	1.20
PDP (fJ)	312.6	18.5	11.2	12.2	15.7
$\lvert MED\rvert$ ($\times 10^{-5}$)	0.16	0.82	1.32	1.23	1.24
$\lvert MRED\rvert$ ($\times 10^{-2}$)	0.82	0.50	4.33	3.47	3.47
$MAED$ ($\times 10^{-2}$)	0.52	1.85	2.75	2.61	2.61
$MARED$ ($\times 10^{-2}$)	0.82	2.93	4.33	4.12	4.12
PSNR (dB)	32.05	29.67	24.32	24.91	24.69
NN Accuracy (%)	97.68	**97.66**	97.62	97.64	97.65

Notes: AppM indicates the use of FP754 or FPNP2 formats and pieces of piece-wise functions used for LA and ALA.
FPLM [21]: LA uses two-piece approximation based on FPNP2 for both two inputs (denoted by N2-N2); ALA uses two-piece approximation (denoted by 2).
LAM [20]: LA uses one-piece approximation based on FP754 for both two inputs (denoted by I1-I1); ALA uses two-piece approximation (denoted by 2).
PWLM1: LA considers FP754 format, but with one input using one-piece approximation and another using two-piece approximation (denoted by I1-I2); ALA uses two-piece approximation (denoted by 2).
PWLM2: LA uses two-piece approximation, but with one input approximated using FP754 and another approximated using FPNP2 (denoted by N1-N2); ALA uses three-piece approximation (denoted by 3).

implemented as $M'_{AB}[q]$. Therefore, the implementation of the exponent computation mainly depends on the use of FP754 or FPNP2. Typically, the use of FPNP2 leads to a larger circuit.

As shown in Table 5, the example designs are compared with two recent LFPMs, LAM [20] and FPLM [21], with respect to the error metrics and hardware performance. They are evaluated from three FP precision levels: 32-bit single-precision, 16-bit half-precision and Bfloat16 format [29] respectively in the form of (1, 8, 23), (1, 5, 10) and (1, 8, 7) bits for the sign, exponent and mantissa, respectively. The Exact FP multiplier (EFM) is considered as the baseline. Note that LAM [20] and FPLM [21] can be generated using the proposed approximation methodology. The four LFPMs were implemented in Verilog and an EFM was obtained using the Synopsys DesignWare IP library (DW_fp_mult). All of the designs were synthesized using the Synopsys Design Compiler (DC) for STM's CMOS 28-nm process and were evaluated with 250-MHz clock frequency. Moreover, to simulate data in DSP and NNs, error results are obtained by considering random FP numbers with standard normal distribution [25]. In the single-precision format, PWLM1 requires a 51× smaller PDP and a 15× smaller area compared to EFM. PWLM1 and PWLM2 are more accurate than LAM with up to 23.7% smaller MRED, but with larger hardware costs. PWLM1 and PWLM2 show up to 39% and 34% smaller PDPs, respectively, than FPLM with lower accuracies as the trade-off. In the half-precision and Bfloat16, PWLM1 and PWLM2 consume at least 1.4× and 1.2× smaller PDPs, respectively, than FPLM. Note that the error metrics for PWLM1 that are shown as identical to PWLM2 are smaller by less than 0.01.

5.2 Applications

This section assesses the capabilities of PWLMs in two error tolerant applications of JPEG compression and NN training.

5.2.1 JPEG Compression

As a widely used lossy compression algorithm, JPEG compresses digital images using the discrete cosine transform (DCT). In this experiment, we perform the JPEG compression of two standard test images including "Lena" and "cameraman". The 256×256 image pixels in the spatial domain are first converted into the frequency domain. The quality level for the compression is set to 50 in the experiment. Then, the high-frequency information is then discarded by using the standard quantization matrix. The compressed image is reconstructed based on de-quantization and the inverse DCT (IDCT).

The multiplications in the DCT and IDCT were implemented by using the EFM and LFPMs. The qualities of the decompressed images using different multipliers compared with the original uncompressed images are assessed by the metric of the peak signal noise ratio (PSNR), as shown in Table 5. A higher precision leads to a higher quality of the decompressed image and the half-precision suits better for JPEG compression compared with Bfloat16 format. The experiment results also suggest that the quality deterioration of images processed by LFPMs measured by the PSNR can be estimated by accuracies of LFPMs indicated by error metrics for all 32-bit and 16-bit precisions. The PSNR values obtained by using PWLM1, PWLM2, and LAM are similar to each other due to their similar $MAED$ and $MARED$ results. Therefore, $MAED$ and $MARED$ are more accurate to predict the PSNR result for JPEG compression compared with the other two error metrics.

5.2.2 Neural Networks

The conventional FP multiplier is replaced with approximate designs in both the training and the inference phases of an MLP by using the Pytorch framework [30]. The same multiplier is used for both the training phase and inference. The MNIST [31] and an MLP network of (784, 128, 10) model were considered for comparison. For a fair evaluation, all the results are obtained under the same experimental setup.

As shown in Table 5, both PWLM1 and PWLM2 increase the classification accuracy compared to the NN with EFM in single-precision and PWLM1 achieves a higher accuracy in half-precision compared to the FPLM and LAM. The Bfloat16 is advantageous over half-precision; moreover, the use of FPLM achieves the highest classification accuracy among all the LFPMs. The classification accuracy in NNs is not strictly consistent with the results in error metrics. A more accurate multiplication result does not always lead to a higher classification accuracy. Therefore, it is necessary to utilize the proposed library of PWLMs to explore the relationship between different error metrics and classification accuracy for better use of approximate FP multipliers in NN training.

An artificial neuron in the Bfloat16 format was implemented to assess the overall hardware cost of NNs since the FP multiplier designs perform well for classification with low energy in this format. The FP adder used in the neuron was obtained using the Synopsys DesignWare IP library (DW_fp_add). The simulation results in Table 6 were obtained at a clock frequency of 125 MHz. It is shown that a neuron using the proposed PWLM1 and PWLM2 show up to 3.9× and 3.7× smaller PDPs, respectively, than the neuron using EFM.

TABLE 6: Circuit Assessment of the Artificial Neuron in the Bfloat16 Format

	EFM	FPLM [21]	LAM [20]	**PWLM1**	**PWLM2**
AppM	-	N2-N2-2	I1-I1-2	I1-I2-2	I2-N2-3
Power (μW)	144.2	59.1	54.8	56.5	57.8
Area (μm^2)	1075.1	537.1	486.1	505.2	514.5
Delay (ns)	6.5	4.4	4.2	4.2	4.3
PDP (fJ)	941.6	263.3	230.2	237.3	248.5

6 Conclusion

In this paper, an approximation design framework using the piece-wise function is proposed for generating logarithmic FP multipliers with various accuracies and hardware designs. The proposed logarithmic FP multipliers are based on two FP number representation formats, the IEEE 754 Standard FP Format and the Nearest Power of Two FP Format. Consider both logarithm and anti-logarithm, the applicable regions are divided into several intervals; two approximation methods with positive or negative errors are then respectively considered for each interval. Both the LA and ALA can be simply implemented by shifting and multiplexing operations. Various configurations of approximation lead to approximate FP multiplier designs with different error characteristics, which support an assessment of error sensitivities of applications. The generic circuit design for the proposed logarithmic FP multipliers is discussed with some simplification methods. Two example designs achieve up to a 51× smaller PDP and a 15× smaller area compared to the exact FP multiplier, while being 23.7% more accurate than Mitchell's logarithmic FP design. Their applications in NN training show high classification accuracy. The proposed library of logarithmic FP multipliers provides potential solutions for efficient NN training through analyzing its inherent error tolerances for different error metrics.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326). T. Zhang was supported by a Ph.D. scholarship from the China Scholarship Council.

References

[1] O. Gustafsson and N. Hellman, "Approximate floating-point operations with integer units by processing in the logarithmic domain," in *ARITH*, 2021, pp. 45–52.

[2] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," *NIPS*, vol. 31, 2018.

[3] F. Kainz, R. Bogart, and D. Hess, "The openexr image file format," *ACM SIGGRAPH Technical Sketches*, 2003.

[4] M. Segal and K. Akeley, "The opengl® graphics system: A specification (version 4.6 (core profile)-may 5, 2022)."

[5] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *ETS*. IEEE, 2013, pp. 1–6.

[6] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proc. IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[7] D. Peroni, M. Imani, and T. S. Rosing, "Runtime efficiency-accuracy tradeoff using configurable floating point multiplier," *TCAD*, vol. 39, no. 2, pp. 346–358, 2018.

[8] V. Camus, J. Schlachter, C. Enz, M. Gautschi, and F. K. Gurkaynak, "Approximate 32-bit floating-point unit design with 53% power-area product reduction," in *ESSCIRC*. IEEE, 2016, pp. 465–468.

[9] M. Franceschi, A. Nannarelli, and M. Valle, "Tunable floating-point for artificial neural networks," in *ICECS*. IEEE, 2018, pp. 289–292.

[10] P. Yin, C. Wang, W. Liu, E. E. Swartzlander, and F. Lombardi, "Designs of approximate floating-point multipliers with variable accuracy for error-tolerant applications," *J. Signal Process. Syst.*, vol. 90, no. 4, pp. 641–654, 2018.

[11] M. Imani, D. Peroni, and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," in *DAC*. IEEE, 2017, pp. 1–6.

[12] J. Y. F. Tong, D. Nagle, and R. A. Rutenbar, "Reducing power by optimizing the necessary precision/range of floating-point arithmetic," *TVLSI*, vol. 8, no. 3, pp. 273–286, 2000.

[13] C. Yan, X. Zhao, T. Zhang, J. Ge, C. Wang, and W. Liu, "Design of high hardware efficiency approximate floating-point FFT processor," *IEEE TCASI*, 2023.

[14] X. Sun, N. Wang, C.-Y. Chen, J. Ni, A. Agrawal, X. Cui, S. Venkataramani, K. El Maghraoui, V. V. Srinivasan, and K. Gopalakrishnan, "Ultra-low precision 4-bit training of deep neural networks," *NeurIPS*, vol. 33, pp. 1796–1807, 2020.

[15] P. Lee, "An evaluation of a hybrid-logarithmic number system dct/idct algorithm [image compression applications]," in *ISCAS*. IEEE, 2005, pp. 4863–4866.

[16] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *ICASSP*, 2017, pp. 5900–5904.

[17] T. Zhang, Z. Niu, and J. Han, "A brief review of logarithmic multiplier designs," in *LATS*. IEEE, 2022, pp. 1–4.

[18] B. Xiong, S. Fan, X. He, T. Xu, and Y. Chang, "Small logarithmic floating-point multiplier based on FPGA and its application on MobileNet," *TCAS II*, 2022.

[19] C. Chen, W. Qian, M. Imani, X. Yin, and C. Zhuo, "Pam: A piecewise-linearly-approximated floating-point multiplier with unbiasedness and configurability," *TC*, 2021.

[20] T. Cheng, Y. Masuda, J. Chen, J. Yu, and M. Hashimoto, "Logarithm-approximate floating-point multiplier is applicable to power-efficient neural network training," *Integration*, vol. 74, pp. 19–31, 2020.

[21] Z. Niu, T. Zhang, H. Jiang, B. F. Cockburn, L. Liu, and J. Han, "Hardware-efficient logarithmic floating-point multipliers for error-tolerant applications," *IEEE TCASI*, vol. 71, no. 1, pp. 209–222, 2024.

[22] S. Ullah, S. S. Murthy, and A. Kumar, "SMApproxLib: Library of FPGA-based approximate multipliers," in *DAC*. IEEE, 2018, pp. 1–6.
[23] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *DATE*. IEEE, 2017, pp. 258–261.
[24] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.
[25] M. S. Ansari, B. F. Cockburn, and J. Han, "An improved logarithmic multiplier for energy-efficient neural computing," *TC*, vol. 70, no. 4, pp. 614–625, 2020.
[26] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, no. 4, pp. 512–517, 1962.
[27] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *TC*, vol. 62, no. 9, pp. 1760–1771, 2012.
[28] S. E. Ahmed and M. Srinivas, "An improved logarithmic multiplier for media processing," *J. Signal Process. Syst.*, vol. 91, no. 6, pp. 561–574, 2019.
[29] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil *et al.*, "A domain-specific supercomputer for training deep neural networks," *Commun. ACM*, vol. 63, no. 7, pp. 67–78, 2020.
[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.
[31] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.

Tingting Zhang (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in the College of Electronic and Information Engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2016 and 2019, respectively. She is working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada, since Sep. 2019. Her research interests include approximate computing, Ising computing, combinatorial optimization and nanoelectronic circuits and systems.

Zijing Niu received the B.Sc. in Microelectronic Science and Engineering from Sichuan University, Chengdu, China, in 2018, and the M.Sc. degree in the Department of Electrical and Computer Engineering from the University of Alberta, Edmonton, AB, Canada, in 2023. Her research interests include approximate computing and hardware designs for accelerating deep learning applications.

Jie Han (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from the Delft University of Technology, Delft, The Netherlands, in 2004. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, and novel computational models for nanoscale and biological applications.